基于生成树算法的公交地铁线路选择模型

摘要

本文建立了优化生成树算法模型以解决公交线路选择问题。生成树算法以起始站作为树的根结点，用树的高度的不断增长代表换乘次数的增长，从而发现起始站在不同换乘次数下所能到达的所有站点，通过判断这些站点中是否存在终止站，找到所有从起始站到终止站的所有路径，再根据决策变量找到最优路径。

优化生成树算法是优化后的生成树算法，与原算法的主要区别是树从起始站和终点站两端同时生长。题目中要求从实际情况出发考虑，满足查询者的不同需求，因此我们将决策变量分别设为乘车时间最短、乘车费用最少、换乘次数最少以及综合评价。

题目中所给的信息主要分为两类：每一条公交或者地铁线路的途经站、公交车辆与地铁的票价以及乘客乘车或换乘所需要耗费的时间。题目最终是要根据这些数据以求得所给的始末站的最优乘车线路。选择的步骤应该首先给出从起始站出发到达终点站的所有乘车线路，再根据不同人群的需求，从所有的线路之中选出基于不同评价标准的最优线路。

题目针对三种出行方式（公共汽车，地铁，步行）的不同特点，提出了三个问题。

第一问中，只有公共汽车可供选择。根据现实生活的经验，公交车价格低廉，容易遇到堵车等问题。在乘客只乘坐公交出行时，路线选择的主要因素通常不是价钱和时间，而是换乘次数。基于这种情况，在这一问中，我们建立了以换乘次数最少为第一目标的模型。在模型求解中，我们尝试了三种算法，即尽量不换乘的贪婪算法、优化的生成树算法、计算任意两个站点间经过几次换乘能够到达的算法——Floyd 算法，并比较了各自的计算结果和时空复杂度。发现优化的生成树算法时空复杂度最小，是满足需要的最优算法。

第二问中，引入了地铁这种独具特色的出行方式。地铁是一种快捷的出行方式，选择地铁的乘客，通常把“时间短”看的比“换乘少”更重要，很多人宁愿增加换乘次数，也不愿坐长时间的直达车。这时由不同的需求出发，会得到几种差异很大的结果，因此我们分别针对乘车时间最少、乘车费用最少和换乘次数最少建立了三个模型进行求解，最后还搭建了综合评价的算法以适应需求不明确的用户，同时做出了相应的评价。

第三问中，由于我们已经知道了所有站点之间的步行时间，同时根据日常生活的经验发现，人们常常不选择频繁换乘，而是选择下了车步行去一个较近的站点换乘，这样减少了换乘次数，同时也减少了由于换乘所耗费的时间和金钱。所以我们人工设立一个邻近站点的时间上确界，以权衡两个站点之间是否可以选择步行而不是换乘。同时将邻近站点加入到搜索域进行扩展搜索，从而选择出包含公交、地铁和步行的最佳路线。

最后我们结合现实生活中的公交选择问题对本题做了扩展。提出并分析了蚂蚁算法，该算法主要是根据统计人们以往选择情况来决定最优路径。

关键字：贪婪算法 生成树 Floyd 公交 地铁 路线选择 换乘 时间 费用 步行

目录

问题重述

我国人民翘首企盼的第 29 届奥运会明年 8 月将在北京举行，届时有大量观众到现场观看奥运比赛，其中大部分人将会乘坐公共交通工具（简称公交，包括公汽、地铁等）出行。这些年来，城市的公交系统有了很大发展，北京市的公交线路已达 800 条以上，使得公众的出行更加通畅、便利，但同时也面临多条线路的选择问题。针对市场需求，某公司准备研制开发一个解决公交线路选择问题的自主查询计算机系统。

为了设计这样一个系统，其核心是线路选择的模型与算法，应该从实际情况出发考虑，满足查询者的各种不同需求。请你们解决如下问题：

1、仅考虑公汽线路，给出任意两公汽站点之间线路选择问题的一般数学模型与算法。并根据附录数据，利用你们的模型与算法，求出以下 6 对起始站→终到站之间的最佳路线（要有清晰的评价说明）。

(1)、S3359→S1828　(2)、S1557→S0481　(3)、S0971→S0485

(4)、S0008→S0073　(5)、S0148→S0485　(6)、S0087→S3676

2、同时考虑公汽与地铁线路，解决以上问题。

3、假设又知道所有站点之间的步行时间，请你给出任意两站点之间线路选择问题的数学模型。

问题分析

随着城市规模的不断扩大，从一个地方到另一个地方，很有可能难以直达，乘客必须换乘才可到达目的地。本题研究的正是城市公交系统的线路选择策略。

这一问题与图论中的最短路问题有些相似，但不完全相同。这是由公交线路的使用者——乘客——的实际需求决定的。

那么，乘客出行的需求到底受什么因素影响呢？这需要对一般乘客出行时心理、动机和目的进行仔细的分析，以确定模型的优化目标和约束条件。在图论中，两点之间的最优路径往往是权值最短的路径，而研究表明，在大多数乘客心目中，最短距离并不是决定公交线路选择的主要因素。

乘客对线路的选择往往受到以下几个因素的影响：

1) 换乘次数，是指乘客从起点到终点的过程中，一共乘坐了几辆公交车；
2) 出行耗时，指乘客从起点到终点的过程中所需的时间。
   包括乘车时间，和换乘时的步行和等候时间；
3) 出行费用，指的是乘客为完成一次出行过程中所支付的各种车费的和。

不同乘客对于各项因素的重视程度是不同的。有些人优先考虑较少的换乘次数，有些人则优先考虑较低的出行耗时。为满足乘客的不同需求，我们同时考虑了“乘车时间短、乘车费用低和换乘次数少”这三种要求。

根据以上分析，模型的目标确立较为简单，而怎样在数量庞大的行车路线和站点间选择出符合各种目标的路线，是问题的关键。

题目中的交通信息分为三类：

1．公交车和地铁的详细线路。

2．公交车与地铁的票价详情。

3．乘客乘车或换乘所需要耗费的时间。

题目针对三种出行方式（公共汽车，地铁，步行）的不同特点，提出了三个问题。

第一种情况，只有公共汽车可供选择。根据现实生活的经验，公交车价格低廉，容易遇到堵车等问题。在乘客只乘坐公交出行时，路线选择的主要因素通常不是价钱和时间，而是换乘次数。基于这种情况，在这一问中，我们建立了以换乘次数最少为目标的模型。

第二问和第三问中，引入了地铁和步行两种各具特色的出行方式。地铁是一种快捷的出行方式，选择地铁的乘客，通常把“时间短”看的比“换乘少”更重要，很多人宁愿增加换乘次数，也不愿坐长时间的直达车。而步行扩大了换乘时的选择范围。这时由不同的需求出发，会得到几种差异很大的结果，因此我们分别针对乘车时间最少、乘车费用最少和换乘次数最少建立了三个模型进行求解。

模型求解的基本步骤是，找到起始站和终点站之间所有可能的线路，根据不同的需求，从中选出相应的最优线路。

问题一

基于最小换乘的公交线路选择模型

1 模型分析：

根据人们的出行习惯，在选择从 A 站到 B 站的行车线路时，首先会先看经过 A 站的车是否有直接到 B 站的，如果有，马上会选择直达车，如果存在不止一条的直达线路，再综合考虑距离、时间、费用等因素，决定乘车方案。

首先考虑一次换乘的乘车方案：即经过 A 站的车与经过 B 站的车有公共站点 C 吗? 如果有，则可以在公共站点 C 处转车；如果没有则要考虑二次换乘的乘车方案，如图 1 所示；如果没有，则需要三次换乘或三次以上才可到达目的地。

选择公交方式出行时，可行的乘车方案一般有多个，出行者必须做出选择。在问题分析中，我们已经论述了，在这里出行者一般以换乘次数为优先考虑的目标 ，公交网络的设计也以减少平均换乘次数为重要目标 ，因此本问以最小换乘次数为第一目标。

另外，旅途时间和旅行费用也是行者关心较多的问题，所以本问选择最短时间为第二目标，最省费用为第三目标。即：在所有可行路径中选择换乘次数最小，且在满足此目标条件下使时间最短的条件下费用最省的路径为最优路径。

由以上原则，选择“换乘次数，时间，费用”字典序的排列的最小值作为目标函数。

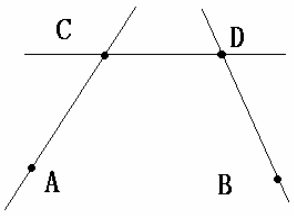


图 1

2 模型假设：

1. 假设乘客在起始公交汽车站的等候时间为零。
2. 假设没有随机突发因素影响路况而使得出行时间加长

3 符号说明：

V_0 起始站点

V_λ 终止站点

p_{ij} 换乘 j 次可以到达的线路 i

v_{ij} 换乘 j 次可以到达的站点 i

N_i 换乘次数

T_i 出行花费的时间

TR 出行路线的集合

$W^{(k)}$ 时间矩阵：存放各站之间相互到达所需要的时间

$R^{(k)}$ 站点路由矩阵：存放各站之间相互到达所需要经过的路由

注：（名词解释）

路由：为一条信息选择最佳线路传输路径的过程，此处即表示站点之间相互到达的路线。

4 模型建立：[1]

设给定起点 V_0 和终点 V_λ，可行的公交路径集合为 $TR=\{TR \mid TR_i =< v_0, p_{i1}, v_{i1}, p_{i2}, v_{i2}, \ldots, v_\lambda >\}$，$TR_i$ 表示在起点 v_0 选择线路 p_{i1} 到达 v_{i1} 换乘 p_{i2} 到达 v_{i2} ，… ，最终到达 v_λ 该路径换乘次数 N_i，所花费的时间 T_i。

出行者的目标函数为：

$$\min_{TR_i} U(N_i, T_i)$$

目标函数性质：

$U(N_i, T_i)$ 是“换乘次数，时间”的字典序函数。这样目标函数可以充分满足在模型分析中的线路选择原则。

5 模型求解：

解法一：（尽量不换乘的贪婪算法）

步骤 1：输入乘车的起始站点 V_0 和目的站点 V_λ。

步骤 2：搜索公交数据库，经过起始站点 V_0 的公交线路存为 p_{i1} (i=l，2，3，… ，m，m 为正整数)，经过目的站点 V_λ 的公交线路存为 $p_{j\lambda}$ (j=1，2，3，…，n，n 为正整数)。

步骤 3：判断是否有 $p_{i1} = p_{j\lambda}$，若满足条件，则将 $TR=\{TR_{i1} \mid TR_{i1} =< v_0, p_{i1}, v_1 >\}$ 并入集合 Z_0。若 $Z_0 \neq \phi$，则公交线路 p_{i1} 即为从站点 V_0 到站点 V_λ 的直达线路。遍历所

有的 p_{i1}。

步骤 4：搜索公交数据库，将公交线路 p_{i1} 所包含的公交站点存为公交换乘矩阵 $O(i,u)$ (u=l，2，3，…，g，g 为正整数)，公交线路 $p_{j\lambda}$ 所包含的站点存为公交换乘矩阵 $P(j,v)$ (v=l，2，3，…，h，h 为正整数)。

步骤 5：判断是否有 $O(i,u)=P(j,v)$，将满足条件的 $O(i,u)$ 记为 v_{i1}，把 $TR=\{TR_{i2}\mid TR_{i2}=<v_0,p_{i1},v_{i1},p_{j\lambda},v_\lambda>\}$ 存入集合 Z_1，若 $Z_1\neq\phi$，则站点 v_{i1} 为从站点 V_0 到站点 V_λ 的一次换乘站点，公交线路 p_{i1}、$p_{j\lambda}$ 为换乘一次的路线，遍历所有的 $O(i,u)$。

步骤 6：搜索公交数据库，将经过站点 $O(i,u)$ 的公交线路存为 p_{i2} (i=1，2，3，…，p，p 为正整数)，公交线路 p_{i2} 所包含的站点 $O_1(k,t)$ (t=1，2，3，…，g，g 为正整数)扩充到公交换乘矩阵 $O(i,u)$ 中。

步骤 7：判断是否有 $O_1(k,t)=P(j,v)$，将满足条件的 $O_1(k,t)$ 记为 v_{i2}，把 $TR=\{TR_{i2}\mid TR_{i2}=<v_0,p_{i1},v_{i1},p_{i2},v_{i2},p_{j\lambda},v_1>\}$ 存入 Z_3，若 $Z_3\neq\phi$，则站点 $O_1(k,t)$ 为从站点 V_0 到站点 V_λ 的二次换乘站点，公交线路 p_{i1}、p_{i2}、$p_{j\lambda}$ 为换乘二次的最优路线，遍历所有的 $O_1(k,t)$。

步骤 8：设定换乘次数的上界 N．若换乘次数小于 N，则转到步骤 6 继续运算。否则转到步骤 9。

步骤 9：令 $Z=Z_0\cup Z_1\cup Z_2\cup\ldots\ldots\cup Z_n$。计算 Z 中每条路线所花费的时间，选择用时最少的路线为最优路径，并输出。

以上步骤如果没有找到合适的公交线路，则输出“没有找到换乘次数不超过 N 次的最优换乘方案”，结束运算。

解法二：（基于树的生成和传递算法及其优化）

基于树的生成和传递算法（简称生成树算法）：

输入：起始站，终点站

初始状态：换乘次数 n=0，路线选择方案集合 $\phi=\{\}$。

步骤 1：根据各个公交车线路及其途经的各站，可以得到经过每个车站的公交车号。

步骤 2：以起始站为根结点来构造树，树的第一层孩子为起始站不需换乘就可以直达的所有站，树的枝干为父子两点之间能够直达的线路名。n=1。（根节点为第 0 层）

步骤 3：如果第 n 层存在终点站的名字，那么树的根结点至叶结点的完整路径就是起始站至终点站的一个乘换车解决方案，并执行步骤 5。

步骤 4：若没有经过终点站的线路，广度优先遍历 n 层所有结点，分别找到能从当前结点直接到达的所有车站名称（如果找到的是已经存在在生成树中的车站，则忽略），作为 n+1 层的孩子，树的枝干为父子两点之间能够直达的线路名。n=n+1。回到步骤 3

步骤 5：设当前路径代表的选择方案为 α，则 $\phi=\phi\cup\alpha$ 。

步骤 6：对于集合 ϕ 中乘换车次数相同的不同出行解决方案，可由线路站点关系表中的站点序号来计算每个方案的出行时间总长度，最后根据时间长度最短原则，从多条候选线路中选出前几条最佳路径， 供乘客决策。[2]

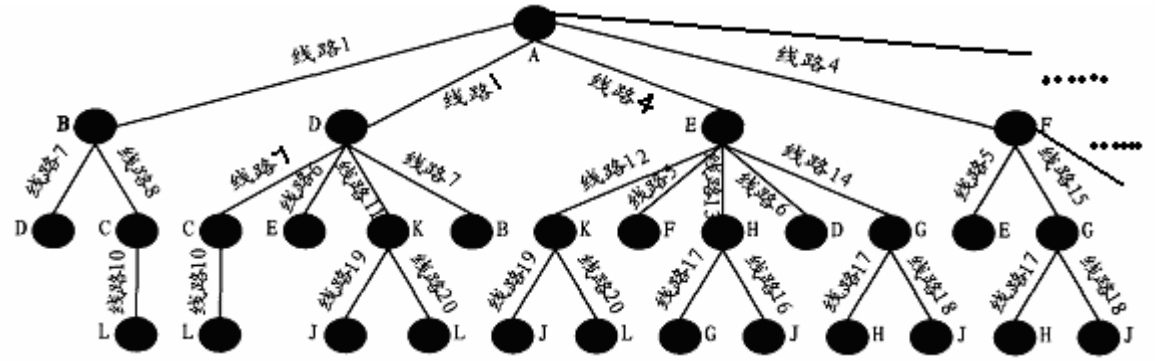


图 2　一般的生成树

生成树算法的优化:

从图 2 中可以明显看出,随着换乘次数的增多,传递算法需要的存储空间急剧增大。造成这种现象的原因是，树中相当一部分车站，只能由起始站到达，并不能到达终点，不是我们需要的换乘站。

解决这个问题的方法如图 3 所示，不仅仅以起始站为根节点构造生成树，同时从终点站开始构造生成树。最终在某一层车站处求交集输出。由于一般的生成树算法的时间复杂度是逐层递增的，树的层数每多一，时间和空间复杂度就多一倍，比上一层远远加大了。从两个方向同时构造生成树，使得存储空间和处理时间随着换乘次数增加而增加的速度大大减缓了。

如下图可知，在相同的大数据量条件下，同样换乘 2 次的情况，图 2 的时空复杂度远远小于图 1 的。可见该算法比原有的普遍意义上的生成树算法优化。

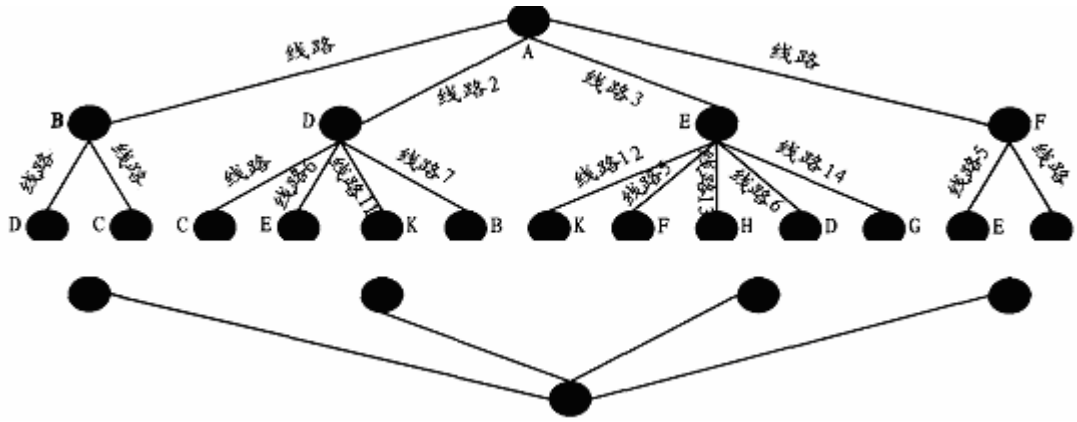


图 3　　改进的生成树

解法三:（所有站点间两两换乘方案——Floyd 算法）

定理 1 对于图 G，如果 $\omega(i,j)$ 表示端 i 和 j 之间的可实现的距离，那么 $\omega(i,j)$ 表示端 i 和 j 之间的最短距离当且仅当对于任意 i，k，j，有 $\omega(i,j) \le \omega(i,k)+\omega(k,j)$。（定理的证明见附录 1）。

Floyd 算法介绍如下：

给定图 G 及其边 (i,j) 的权 $\omega_{i,j}(1\le i\le n,\quad 1\le j\le n)$，则

F_0：初始化时间矩阵 $W^{(0)}$ 和站点路由矩阵 $R^{(0)}$ 为 $W^{(0)}=\left[\omega_{i,j}^{(0)}\right]_{n\times n}$，$R^{(0)}=\left[r_{i,j}^{(0)}\right]_{n\times n}$

其中

$$\omega_{i,j}^{(0)}=\begin{cases}\omega_{i,j} & e_{i,j}\in E\\ \infty & e_{i,j}\notin E\\ 0 & i=j\end{cases}$$

$$r_{i,j}^{(0)}=\begin{cases}j & \omega_{i,j}^{(0)}\neq\infty\\ 0 & \text{其他}\end{cases}$$

F_1：已求得 $W^{(k-1)}$ 和 $R^{(k-1)}$，依据下面的迭代求 $W^{(k)}$ 和 $R^{(k)}$：

$$\omega_{i,j}^{(k)}=\min\left\{\omega_{i,j}^{k-1},\omega_{i,k}^{(k-1)}+\omega_{k,j}^{(k-1)}\right\}$$

$$r_{i,j}^{(k)}=\begin{cases}r_{i,k}^{(k-1)} & \omega_{i,j}^{(k)}<\omega_{i,j}^{(k-1)}\\ r_{i,j}^{(k-1)} & \omega_{i,j}^{(k)}=\omega_{i,j}^{(k-1)}\end{cases}$$

F_2：若 $k<n$，重复；$k=n$，终止。

算法要执行 n 次迭代，第 k 次迭代的目的是允许经过第 k 个站点换乘，考察是否可以使各个站点之间相互到达的时间缩短。前 k 次迭代的目的是允许经过第 1，2，…，k 个站点换乘，考察是否可以使各个站点之间相互到达的时间缩短。容易估计 Floyd 算法的计算量是为 $O(n^3)$。

对于本题来说，如果两个站点之间有车直达，相应权值为 1；无车直达，权值为 0。至此，我们建立了一个更加容易求解的模型。

求解步骤

步骤 1：整理题目所给的车站信息，将每个车站所经过的车组成该车站的集合。

步骤 2：建立一个邻接矩阵，表示每两个车站之间是否有直达车，用 1，0 表示。可以通过步骤 1 中两个车站的集合是否有交集来判断。建立一个三维矩阵，第三维的第零个元素表示两个车站间的公交车总数，之后的元素分别表示两个车站间经过的直达车的线路号。例：$\mathrm{A}=\begin{bmatrix}\mathbf{1}&\mathbf{0}&\mathbf{1}\\ \mathbf{1}&\mathbf{1}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{1}\end{bmatrix}$

步骤3：判断起始站和终点站之间是否有直达车，所对应的邻接矩阵中的元素是否为1，若为1，从三维矩阵中找出直达车即可；若不为1，即不能直达，则通过对直接关系矩阵的计算，可以得到更多的路线信息.

布尔乘法和布尔加法的运算法则为：

0+ 0= 0，　1+ 0=1，　1+ 1=1，

0*0 = 0，　1*0 =0，　1*1 =1，

可以得到：$A^2 = A*A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$，$A^2 - A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$即表示经过一次换乘可到达的两站，从站2到站3须经过一次换乘才能到达。

步骤4：现在需要做的就是找出经过换乘才能到达的两个车站如何找到之间的线路。还是前面的例子：$A^2 - A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$，$a^2_{23} = 1, a_{23} = 0$，

$\because a_{ij}^{\ 2} = \sum_{k=1}^{n} a_{ik} \times a_{kj}$，只须找出$a_{ik} \times a_{kj} = 1$，表示从i到k站，再从k到j站。

$a_{21} \times a_{13} = 1$，只须从三维矩阵中分别找到2站到1站和3站到2站的线路。同样的，对于需要换乘两次，三次也可用类似的方法求得。特别的，

$A^3 = A^2 * A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$，上式表明再计算下去，已没有意义，此矩阵称为可达矩阵，计算到此为止。一般对于大数据量的公交矩阵，换乘3次以上基本可以达到任意两站之间均有路径，即变为所有元素为1的矩阵。

步骤 5：计算出给出的每条路线所花费的时间，选择用时最少的路线为最少路径，并输出其线路。

6 求解结果：

（1）$S3359 \rightarrow S1828$

用时最短为 101 分钟，费用为 3 块钱，最优线路为：

坐 L436 号车（下行）在 S1784 转 L167 号车（下行）可达(一次换乘)
坐 L436 号车（下行）在 S1784 转 L217 号车（下行）可达(一次换乘)

（2）$S1557 \rightarrow S0481$

用时最短为 100 分钟， 费用为 5 块钱，最优线路为：

坐 L363 号车（下行）在 S0055 转 L348（下行）号车在 S2361 转 L312（下行）号车可达(二次换乘)

（3）$S0971 \rightarrow S0485$

用时最短为 128 分钟， 费用为 4 块钱，最优线路为：

坐 L013 号车（下行）在 S2184 转 L0417（下行）号车可达(一次换乘)

（4）$S0008 \rightarrow S0073$

用时最短为 83 分钟， 费用为 2 块钱，最优线路为：

坐 L159 号车（下行）在 S0400 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2633 转 L474 号车（上行）可达(一次换乘)

坐 L159 号车（下行）在 S3053 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2683 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S0291 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S3614 转 L058 号车（下行）可达(一次换乘)
坐 L355 号车（下行）在 S3917 转 L345 号车（上行）可达(一次换乘)
坐 L355 号车（下行）在 S2303 转 L345 号车（上行）可达(一次换乘)

（5）$S0148 \rightarrow S0485$

用时最短为 79 分钟， 费用为 4 块钱，最优线路为：

坐 L308 号车（上行）在 S2221 转 L481（上行）号车在 S2027 转 L007（上行）号车可达(二次换乘)

（6）$S0087 \rightarrow S3676$

用时最短为 65 分钟， 费用为 2 块钱，最优线路为：

坐 L454 号车（上行）在 S3496 转 L209 号车（下行）可达(一次换乘)

模型评价：

文中三个算法模型都是较优解，无法得到和确定最优解，下面给出了一组算法效率对比数据：（表格里“—”表示在可接受的时间范围内解不出来）

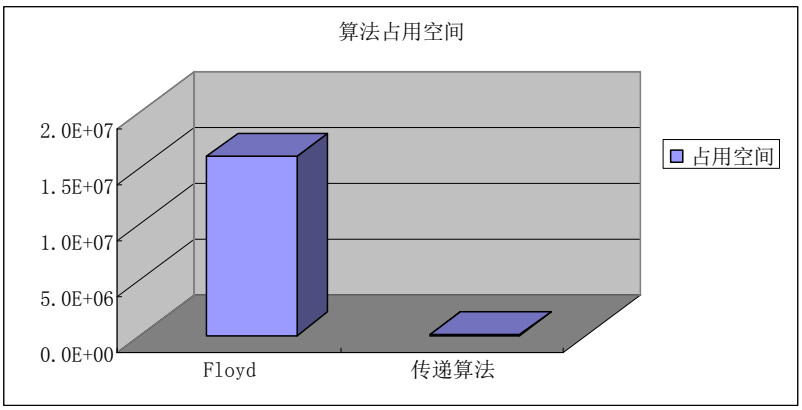
线路 / 时间	直达	一次换乘	两次换乘	三次换乘
1	小于1秒	2秒	—	—
2	小于1秒	2秒	2分钟	3.5分钟
3	40分钟	—	—	—

我们发现，解法一（贪婪算法）在计算直达以及一次换乘的公交路线是可以接受的，计算多于一次换乘的时间复杂度太高，运行时间过长，被我们放弃。解法一的优点是与人的思维方式类似，容易理解。缺点为需要的存储空间大，时间复杂度高到了不可接受的程度。

解法二（生成树算法）可以求出直达，换乘一次~三次的公交路线选择结果，且运行较快，算法也较直观，空间和时间上适中，我们认为是目前为止最好的模型

解法三（Floyd 算法）的时间复杂度最高，对空间的要求也最大，具体的分析属于计算机科学的范畴，这里只画出生成树算法和 Floyd 算法占用空间的比例（图 4）。

Folyd 算法也有其优点，它能够求出任意两个站点之间需要换乘的次数，且算法实现非常简单（只有矩阵的相乘），适合大规模的计算机运算。



（图 4）

问题二

模型分析

在问题一求解的基础上，同时考虑公汽与地铁线路。地铁在这里起的作用有两点，一来可以让同一地铁站对应的任意两个公汽站之间可以通过地铁站换乘（无需支付地铁费），这样可以在一定程度上减少换乘次数；二来可以让公汽换乘地铁，同样，也可以减少换乘次数。因此第二问的题设条件使得公交线路的选择范围大大增加了。

模型假设

1. 地铁不分上下行，任何两个地铁站点之间都可以到达。
2. 若换乘 3 次还到达不了终点站，我们则认为其相互之间不可达。

新增符号说明

$v_{ij}^{'}$ 地铁站点

$p_{ij}^{'}$ 地铁线路

模型建立：

设给定起点 V_0 和终点 V_λ，可行的公交路径集合为 $TR=\{TR \mid TR_i =< v_0, p_{i1}, v_{i1}, p_{i2}, \ldots p_{ij}^{'}, v_{ij}^{'}, \ldots, v_\lambda >\}$，$TR_i$ 表示在起点 v_0 选择线路 p_{i1} 到达 v_{i1}，…换乘 $p_{ij}^{'}$ 到达 $v_{ij}^{'}$，…，最终到达 v_λ 该路径换乘次数 N_i，所花费的时间 T_i。

出行者的目标函数为：$\min_{TR_i} U(T_i)$

目标函数性质：$\dfrac{\partial U}{\partial N_i} < 0$

算法一：（生成树算法的改进）

步骤 1：先判断起始点是不是地铁边上的公交站点，若不是，则把起始站点可以直达的所有公车站点放入集合 1。若是，则把起始站点可以直达的所有公车站点放入集合 1，同时将所有地铁站所邻近的公车站放入集合 1。

步骤 2：对集合 1 里每个元素进行步骤 1 的处理，将得到的所有站点放入集合 2。

步骤 3：同理生成集合 3 和集合 4

步骤 4：如果在集合 1 中能够找到终点，就找到了一种直达的方案：起点-终点。同理，如果在集合 2 中能够找到终点，就找到了一种换乘一次的方案：起点-集合 1 中的点-终点。

步骤 5：记录下结果。求出经过目标站的线路。根据乘客对时间、价钱、换乘次数的不同需求，分别运算，最终给出最佳线路。

寻找最近的地铁换乘站：[3]

中字型地铁网可看作单线型与环线型的合成，对应的算法是两者相应算法的结合，关键在于换乘的处理，即假定单线与环线的交点为 cc1，cc2，则确定单线与环线间的换乘地点至关重要。如图 3 所示，假定 k 在单线上，h 在环线上，计算环线距离 LL(cc1，h)，LL(cc2，h)，比较两者的大小，换乘在距离小的相应交点进行。如图 3 所示网络，在 cc1 点换乘，由于 $LL(cc1,h) < LL(cc2,h)$，则 $\mathrm{LL(k,h)} = \mathrm{LL(k,cc1)} + \mathrm{LL(cc1,h)}$，含地铁方式的有效出行路线为 s—k—cc1—h—e，反之，则 $\mathrm{LL(k,h)} = \mathrm{LL(k,cc2)} + \mathrm{LL(cc2,h)}$，含地铁方式的有效出行路线为 s—k—cc2—h—e

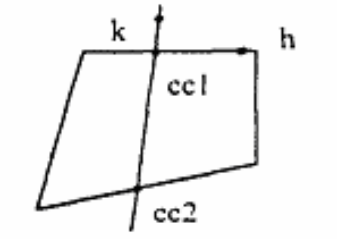


图 3

算法二：（基于时间最优的算法）

算法分析：

正常情况下,地铁的速度往往远快于公交系统,所以基于时间最优的情况下,我们选择有地铁就搭地铁的方案。

求解：

步骤 1：先判断起始点是不是地铁边上的公交站点,若是,则考虑 3 或 4。

步骤 2：找出起始站所连接的所有站,如果有终点站,记录路径,转回第一步,如果没有,则查看这些站有没有地铁边上的站,若有,3 或 4,否则,重复执行 2.

步骤 3：（通过地铁换乘）找一个地铁对应的其它站,若有终点站,转回第一步,若没有,找这些站可直达的所有站点。

步骤 4：（换乘地铁）找地铁沿线的所有车站,若有终点站,记录路径,转回第一步,若没有,则记录这些公交站可直达的所有站,查找终点。

时间优先求解结果：

（1）$S3359 \rightarrow S1828$

用时最短为 99.5 分钟最优线路为：

坐 L436 号车(下行)在 S0618 转地铁 T1 线,在 D08 上车做到 D12 换乘 T2 线做到 D38,在 S3262 换乘 L41 号车（下行）可达(三次换乘)

（2）$S1557 \rightarrow S0481$

用时最短为 92 分钟，最优线路为：

坐 L363 号车（下行）在 S1920 下车在 D20 转地铁 T1 线在 D01 下车在 S0567 转 L448（上行）号车可达(二次换乘)

（3）$S0971 \rightarrow S0485$

用时最短为 128 分钟，最优线路为：

坐 L013 号车（下行）在 S2184 转 L0417（下行）号车可达(一次换乘)

（4）$S0008 \rightarrow S0073$

用时最短为 83 分钟，最优线路为：

坐 L159 号车（下行）在 S0400 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2633 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S3053 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2683 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S0291 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S3614 转 L058 号车（下行）可达(一次换乘)
坐 L355 号车（下行）在 S3917 转 L345 号车（上行）可达(一次换乘)
坐 L355 号车（下行）在 S2303 转 L345 号车（上行）可达(一次换乘)

（5）$S0148 \rightarrow S0485$

用时最短为 58。5 分钟， ，最优线路为：

坐 L308 号车（上行）在 S0302 下车换乘地铁 T1 线，在 D03 上车 D21 下车换乘 L50（上行）号车在 S0466 上车可达(二次换乘)

（6）$S0087 \rightarrow S3676$

用时最短为 33 分钟，最优线路为：

在 S0087 走到 D27 坐地铁 T2 线在 D36 下车走到 S3676 可达(直达)

算法三：（基于费用最优的算法）

算法分析：

相对于模型一而言,地铁虽然快,但它的费用相当于最贵的公交线路,所以基于费用最优点情况下,我们选择公交线路的时候,能不搭地铁就不搭地铁,但利用地铁进行公交换乘,可以较明显得降低换乘次数。

求解:

步骤 1：把通过地铁换车写成一函数,在一问中每次搜到的站点集中遍历一次,查看其在不在地铁边上,若在,就调用换车函数,不在,则照一问的算法进行。

步骤 2：通过地铁换车函数,列出这个站点可到达的所有的站点。

费用优先求解结果：

（1）*S*3359 → *S*1828

费用为 3 块钱，最优线路为：

坐 L436 号车（下行）在 S1784 转 L167 号车（下行）可达(一次换乘)
坐 L436 号车（下行）在 S1784 转 L217 号车（下行）可达(一次换乘)

（2）*S*1557 → *S*0481

费用为 3 块钱，最优线路为：

坐 L84 号车（下行）在 S3389 转 L80（下行）号车在 S1327 转 L72（下行）号车可达(二次换乘)

（3）*S*0971 → *S*0485

费用为 4 块钱，最优线路为：

坐 L013 号车（下行）在 S2184 转 L0417（下行）号车可达(一次换乘)

（4）*S*0008 → *S*0073

费用为 2 块钱，最优线路为：

坐 L159 号车（下行）在 S0400 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2633 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S3053 转 L474 号车（上行）可达(一次换乘)
坐 L159 号车（下行）在 S2683 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S0291 转 L058 号车（下行）可达(一次换乘)
坐 L159 号车（下行）在 S3614 转 L058 号车（下行）可达(一次换乘)
坐 L355 号车（下行）在 S3917 转 L345 号车（上行）可达(一次换乘)
坐 L355 号车（下行）在 S2303 转 L345 号车（上行）可达(一次换乘)

（5）*S*0148 → *S*0485

费用为 4 块钱，最优线路为：

坐 L308 号车（上行）在 S2221 转 L481（上行）号车在 S2027 转 L007（上行）号车可达(二次换乘)

（6）*S*0087 → *S*3676

费用为 2 块钱，最优线路为：

坐 L454 号车（上行）在 S3496 转 L209 号车（下行）可达(一次换乘)

算法四：（综合评价最优的算法）

算法分析：

从起始地V_1到终止地V_λ，公交换乘方案肯定不止一条，如何来判定其优劣呢?常理来说，换乘次数最少是公交乘客出行时考虑的第一重要因素，基于这一因素基础之上，考虑人们出行的各种实际情况分析出各种策略，如时间优先、费用优先、换乘次数优先，景观优先等，这些都是不同的人出行考虑的不同因素，但如果行人也不太清楚自己的需求，那么我们需要给出对路线的一个综合评价。

算法假设

1. 假设行人没有除去时间优先、费用优先、换乘次数优先的其他需求。
2. 假设时间权重为 30%，费用权重为 30%，换乘次数权重为 40%。

新增符号说明

O 综合评价指标

T_t 时间权重

M_m 费用权重

C_c 换乘次数权重

求解：

用以下所述权值来衡量最优的换乘方案：O= 时间× 权重%+费用× 权重%+换乘次数×权重%。

具体实现方法为：

步骤 1：取时间最优的线路$TR=\{TR\mid TR_i=<v_0,p_{i1},v_{i1},p_{i2},v_{i2},\ldots,v_\lambda>\}$的排名前30%，取费用最优的线路$TR'=\{TR\mid TR_i=<v_0,p_{i1},v_{i1},p_{i2},v_{i2},\ldots,v_\lambda>\}$的前 30%，取换乘次数最优的线路$TR''=\{TR\mid TR_i=<v_0,p_{i1},v_{i1},p_{i2},v_{i2},\ldots,v_\lambda>\}$的前 40%，求三者的交集$TR_0$。

若三者没有交集，则重新调整权值，使得$TR=TR\times(1+T_t\%)$，$TR'=TR'\times(1+M_m\%)$，$TR''=TR''\times(1+C_c\%)$。重新运行步骤 1。否则运行步骤 2。

步骤 2：记录TR_0中的每条路线并输出，供行人选择。

模型评价：

广义算法求出的路线选择解可以根据不同的需求再进一步进行筛选，但由于计算复杂度高，运算起来较慢，若做成实际的公交查询系统，用户可能忍受不了运算过程中长时间的等待。

但时间优先算法与费用优先算法则根据具体情况对模型进行求解，由于目的明确，手段清楚，所以计算的时间复杂度大大降低，明显优于广义的算法的时间效率。

而在现实生活中，我们常常需要给一条线路做综合评价。可以任意调整每种因素所占的权重，来计算综合因素 O，然后通过比较 O 来判定多种换乘方案中的最优方案，使得模型结果在现实生活中更加实用。在上述情况中如果存在不止一种的选择方案，则再考虑选择综合因素最高的作为优先的乘车方案。

问题三

基于邻接站点可步行的公交线路选择模型[4]

模型分析：

分析上面的算法可以发现只有当不同线路之间具有公共站点时才能够进行转车，这样计算出来的结果有时并不符合实际情况．比如在实际出行时只需换乘二次便可到达目的地，但计算出来的结果却需要换乘三次或四次。出现这种情况的原因是忽视了现实生活中人们步行小段距离再转车的现象。具体地说，人们在转车时，并不是下车后直接在下车的站点处转车．往往需要步行一小段距离到附近的站点去转车。

用以描述公交站点与邻近站点的距离关系，通常是根据人们的行为习惯和平均公交路段长度来决定的。而在本题中，路段长度用步行时间来描述。邻近站点的存在使得人们在选择换乘路线时多了一个考虑，就是如果在某一站点下车后没有直接换乘的车次．还可以考虑附近的站点是否有换乘车次。

模型假设

1. 所有站点之间的步行时间是固定不变的，不受其他外界随机因素的干扰。
2. 假设只有步行时间$t_{ij} \le T$ (T 为取定的整数)的站点之间才可以互称为邻近站点。
3. 人们只会选择邻近站点或同一站点进行车次的换乘。

新增符号说明

T 判定是否为邻近站点的时间上确界

t_{ij} 站点 i 与其邻近站点 j 之间的步行时间

模型建立与问题一所建立的模型一致

模型求解：

在问题 1 贪婪算法中的步骤 4 中，将$O(i,u)$的邻近站点参与搜索，如果与$P(j,v)$及其邻近站点有相同站点，即可认为通过步行到达其邻近站点换乘到达终点。最终确定时间最优的路线时还要加入步行到邻近站点所花费的时间t_{ij}。

模型评价

这种算法计算出的换乘方案更加符合站点的分布情况以及人们出行时的实际选择情况。

模型扩展

蚂蚁算法：

公交选路在现实中的应用。

因为在现实生活中，公交系统是相当不稳定的，尤其是北京市，堵车简直就是家常便饭。所以人们出行不仅仅考虑的是路线，有的时候还得考虑路况。但是每天都在变化的路况如何定评价标准呢？我们在此提出的蚂蚁算法，就是根据选择走的人的多少，来确定是否选择该路径。可以想见，人们从 A 地到 B 地最常走的就是 C 路线，那么，走 C 路线肯定会比其他路线有更高的效率。以下将稍加描述一下蚂蚁算法：

城市的公交线网可用连通的赋权有向图G(V ，E) 表示。V = { v1 ， v2 ， …， vn} 为顶点集， n 为公交站点(包括场站) ;站点之间的有向连线(单向、双向) 集E 表示公交线路及其走向;连线上对应的权值表示蚂蚁出行路径的激素强度。

已知起、迄点的城市公交出行线网可简化为图形式。

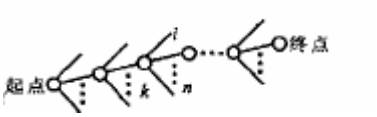


图2　公交出行线网

假设任一公交站点k 的出度为n (即从公交站点k 可乘坐n条公交线路， 即站点k 有n 个目标站点) 。其中起点代表蚁穴， 终点代表食物源。

蚂蚁在出行选择路径时，从起点开始，根据起点站点可乘坐的n 条公交线路的激素强度，随机选择第i 条继续前进达到下一站点k ， 并判断是否达到食物源。若达到则出行成功， 并修改公交线网的激素强度(使蚂蚁按原路返回，增加经过的线路的激素强度，反之则减少) ; 若否， 则出行站点数加1 ， 并继续向前查找，直到搜寻到食物源为止。

参考文献：

[1] 赵巧霞 马志强 张发，以最小换乘次数和站数为目标的公交出行算法， 计算机应用， 第24卷第12期:136-146，2004年

[2] 孙湧，基于宽度优先遍历树的公交线路换乘算法， 深圳职业技术学院学报， 2004年第4期:10-12。

[3] 周刚 王炜，地铁线路客流分配方法与算法研究，广东公路交通，总第70期：32-35，2001年。

[4] 扈震 张发勇 刘书良，城市公交换乘数据模型研究及算法实现， 电信网，2007年4月第4期:71-74

附录：

1、定理 1 对于图 G，如果 $\omega(i,j)$ 表示端 i 和 j 之间的可实现的距离，那么 $\omega(i,j)$ 表示端 i 和 j 之间的最短距离当且仅当对于任意 i，k，j，有 $\omega(i,j) \le \omega(i,k) + \omega(k,j)$

证明：

首先，如果 $\omega(i,j)$ 表示端 i 和 j 之间的最短距离，则 $\omega(i,j) \le \omega(i,k) + \omega(k,j)$。

下面考虑充分性：

若 μ 是任一个从端 i 到端 j 的链， $\mu = (i, i_1, i_2, \ldots, i_k, j)$，则反复应用充分条件，有

$$\begin{aligned}\omega(i,j) &\le \omega(i,i_1)+\omega(i_1,j)\\ &\le \omega(i,i_1)+\omega(i_1,i_2)+\omega(i_2,j)\\ &\le \dots\\ &\le \omega(i,i_1)+\omega(i_1,i_2)+\omega(i_2,i_3)+\dots+\omega(i_k,j)=\omega(\mu)\end{aligned}$$

因为 $\omega(i,j)$ 表示端 i 和 j 之间的可实现的距离，则 $\omega(i,j)$ 表示端 i 和 j 之间的最短距离。